

In the Claims:

Please amend the claims as follows:

1. (previously amended) A method to detect a fault in a CPU of an industrial controller during on-line safety control of real world objects the method comprising:
  - compiling an application program into assembler instructions, which application program was previously defined in a high level language intended for safety control,
  - compiling a test application into assembler instructions where the assembler instructions is a subset of the total number of assembler instructions available for the CPU, which test application was previously defined in said high level language intended for safety control and the test application covers at least all language constructs used in the application program,
  - downloading the application program and the test application to a central unit of an industrial controller,
  - executing repeatedly the test application in the industrial controller,
  - comparing repeatedly by means of a test module a result from the test application with the pre-defined result in the test module,
  - detecting a fault in the CPU as the result from the test application does not equal the pre-defined result stored in the test module and the unexpected result of the test application is due to the execution of an assembler instruction of the test application,
  - aborting the execution of the application program wherein the application program is prohibited from executing the assembler instruction which otherwise would cause the application program to fail.

2. (previously amended) The method according to claim 1, wherein the assembler version of the test application comprise assembler code derived from all language constructs in the high-level language available for safety control of real world objects.

3. (previously amended) The method according to claim 1, wherein the high level language intended for safety control is based on IEC 61131-3.

4. (previously amended) The method according to claim 3, wherein defining a test application analyzing the application in order to determine subset and software libraries used in the said application code.

5. (previously amended) The method according to claim 4, wherein a test application is defined automatically without any additional command from an application programmer.

6. (previously amended) The method according to claim 5, wherein executing the test application repeatedly is performed by a cyclic execution of the test application where the cycle time is determined from a given process safety time value.

7. (previously amended) The method according to claim 6, wherein the test application before an execution receives a set of input values and the input values are generated by means of the test module.

8. (previously amended) The method according to claim 7, wherein down-loading the application program and test application additionally comprise down-loading a predefined result.

9 10. (currently amended) A computer program product, for use in an industrial control system, ~~containing~~ the computer program product comprising:

software code means loadable into the central unit of an industrial controller intended for safety control of real world objects, said computer program product comprising means to make the industrial controller carry out a method comprising:

~~execute~~ repeatedly executing a ~~the~~ test application in ~~the~~ an industrial controller, wherein the repeated execution comprises execution of assembler instruction of the test program,

~~compare~~ repeatedly comparing with ~~by means of~~ a test module a result from the test application with ~~the~~ a pre-defined result in the test module,

detecting ~~detect~~ a fault in ~~the~~ a CPU ~~as the~~ if the result from the test application does not equal the pre-defined result stored in the test module and ~~the~~ an unexpected result of the test application is due to the execution of ~~an~~ the assembler instruction of the test application, and

aborting ~~abort~~ ~~the~~ execution of ~~the~~ an application program wherein the application program is prohibited from executing the assembler instruction which otherwise would cause the application program to fail.

10 11. (currently amended) An industrial control system, comprising an industrial controller with a central unit equipped with a CPU intended for safety control of real world objects, an I/O system wherein the CPU is subject to fault detection according to ~~the~~ a method in ~~claim 1~~ comprising

compiling an application program into assembler instructions, which application program was previously defined in a high level language intended for safety control,

compiling a test application into assembler instructions where the assembler instructions is a subset of the total number of assembler instructions available for the CPU, which test application was previously defined in said high level language intended for safety control and the test application covers at least all language constructs used in the application program,

downloading the application program and the test application to a central unit of an industrial controller,

executing repeatedly the test application in the industrial controller,

comparing repeatedly by means of a test module a result from the test application with the pre-defined result in the test module,

detecting a fault in the CPU as the result from the test application does not equal the pre-defined result stored in the test module and the unexpected result of the test application is due to the execution of an assembler instruction of the test application, and

aborting the execution of the application program wherein the application program is prohibited from executing the assembler instruction which otherwise would cause the application program to fail.